1. Introduction

Goal of this Note:

1. Understand Python Division, Floor Division and Modulo
2. Explore Python on the Jupyter.
   a. Variables
   b. Lists
   c. Conditionals

2. How to Approach Python for the First Time

Recall that Python is a high-level programming language, but fortunately its purpose is to be friendly to beginners. So, here's a helpful tip that might sound childish but is very effective if you have never coded before in your life: pretend that Python is not a program but an unintelligent robot. You can command this robot to perform tasks but since the robot happens to be so doltish, you reduce your commands to generic statements in English that lack grammar.

3. Division, Floor Division and Modulo

True Division: / (decimal division)	Floor Division: // (integer division)	Modulo: % (remainder)
>>> 1 / 5 0.2	>>> 1 // 5 0	>>> 1 % 5 1
>>> 25 / 4 6.25	>>> 25 // 4 6	>>> 25 % 4 1
>>> 4 / 2 2.0	>>> 4 // 2 2	>>> 4 % 2 0
>>> 5 / 0 ZeroDivisionError	>>> 5 // 0 ZeroDivisionError	>>> 5 % 0 ZeroDivisionError

Thanks to CS 61A, we have the wonderful table above that summarizes the functionality of these operations. True Division (/) is the division we carry out normally. Floor division (//) returns the quotient. The modulo(%) returns the modulus of a number i.e., it returns the remainder.

[DEMO on JUPYTER]

4. Variables

Variables are containers that store the data we want to keep track of. In a very *'obnoxious'* manner, variables have types. To find the variable type, we use the type() function where the variable is passed in as an input to the function. Mr. Python – an otherwise doleful robot – spares us with declaring our data types. Sadly, this is no act of mercy. While not declaring our data type can seem like a good thing, the caveat is that our variable type can change dynamically. If I declare a as an integer, I can change it to be a Boolean value in the very next step. So, we have to be a little careful with what data type our variables are. Coming to the data types themselves, refer to the table below to find the different data types:

Data Type	Explanation	Example
Integer (Int)	Positive or negative integer	A = 5
Float	Decimal (floating point) number	A = 3.1415926
Boolean (Bool)	True or False	A = True or A = False
List	Ordered List of Value	A = [1, 2, 3]
String (Str)	A List of Characters (text)	A = 'Marvin the depressed robot'
Dictionary (Dict)	A mapping of keys and values	A = {'e':2.718, 'pi', 3.141}
None	Nonetype (null or missing value)	A = None

5. Print Statement and Operations

Naturally, print statement prints out results. If you are printing a string, enclose them with brackets and put single quotation marks around them for strings or type in the variable name you want to print.

Next, let's review a few basic mathematical operations Python can carry out:

Operator	Name	Example
+	Addition	A + B
-	Subtraction	A - B
*	Multiplication	A*B
**	Exponentiation	A**B
==	Equal	A == B
!=	Not equal	A!=B
>	Greater Than	A > B
<	Less than	A < B
>=	Greater Than or Equal To	A >= B
<=	Less than Or Equal To	A <= B
and	It returns true if both statements are true.

If there is one false statement, returns first False statement. If both are True, returns the last True Statement. | A > B and B > C |
| or | As a condition, it returns true if both statements are true.

If there is one false statement, returns first True statement. If both are True, returns the first True Statement. | A > B or B > C |
| not | If result is True, returns false | not (A < B) |

[DEMO on JUPYTER]

6. Lists

A list is a data structure that acts like a container and stores multiple elements. Each element can be of any type, even a list itself. We write a list as a comma-separated list of expressions in square brackets.

List Elements	1	2	3
Forward Index	0	1	2
Reverse Index	-3	-2	-1

Lists are accessed with their index. Usually, we use forward index, but if you want to reverse your list, you can use reverse index too. Forward Index starts from the start of the list and Reverse Index starts from the back of the list. It is the forward index subtracted by the length of the list

We can also slice lists. Slicing a list creates a copy of part or all of list.

The syntax to slice a list is 'list[<start index>:<end index>:<step size>]'. This expression evaluates to a new list containing the elements of list:
   a. Starting at and including the element at <start index>.
   b. Up to but not including the element at <end index>.
   c. With <step size> as the difference between indices of elements to include.

If the start, end, or step size are not explicitly specified, Python has default values for them. A negative step size indicates that we are stepping backwards through a list when including elements.

There are various methods you can use on List. If you want to add a single element to a python list, you use the append() method and if you want to add multiple elements to a list, you use the extend() method.

[DEMO on JUPYTER]

7. The Go-Do-Your-Homework-Instead Section

1. The modulo is a very interesting operation. There is an entire section of arithmetic called modular arithmetic that exists because of the modulus operation. They are used in encryption and cryptography, for instance the RSA encryption.

2. The print statement actually returns the value None of type NoneType. What you see in a print statement is the outcast/side-effect. It's this side-effect that makes the print function a non-pure function. We will learn more about this next week. This is also why functions use the return statement to return a value and do not use print statements to return values.

3. Lists are not the only types of containers. We also have linked lists where the zeroth index of the list is the head and the first index either points to another list or is empty/null. They are probably my most favorite topic in Python. I strongly encourage you to read about them if you have the time. You can also simply send me a message on Slack if you want more info on Linked List.